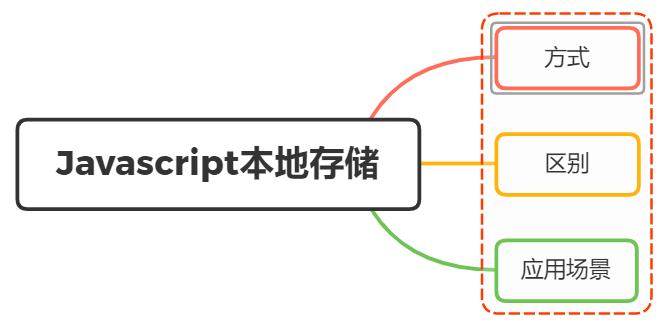
# 面试官：Javascript本地存储的方式有哪些？区别及应用场景？



## 一、方式

javaScript本地缓存的方法我们主要讲述以下四种：

* cookie
* sessionStorage
* localStorage
* indexedDB

### cookie

Cookie，类型为「小型文本文件」，指某些网站为了辨别用户身份而储存在用户本地终端上的数据。是为了解决 HTTP无状态导致的问题

作为一段一般不超过 4KB 的小型文本数据，它由一个名称（Name）、一个值（Value）和其它几个用于控制 cookie有效期、安全性、使用范围的可选属性组成

但是cookie在每次请求中都会被发送，如果不使用 HTTPS并对其加密，其保存的信息很容易被窃取，导致安全风险。举个例子，在一些使用 cookie保持登录态的网站上，如果 cookie被窃取，他人很容易利用你的 cookie来假扮成你登录网站

关于cookie常用的属性如下：

* Expires 用于设置 Cookie 的过期时间

Expires=Wed, 21 Oct 2015 07:28:00 GMT

* Max-Age 用于设置在 Cookie 失效之前需要经过的秒数（优先级比Expires高）

Max-Age=604800

* Domain指定了 Cookie 可以送达的主机名
* Path指定了一个 URL路径，这个路径必须出现在要请求的资源的路径中才可以发送 Cookie 首部

Path=/docs # /docs/Web/ 下的资源会带 Cookie 首部

* 标记为 Secure的 Cookie只应通过被HTTPS协议加密过的请求发送给服务端

通过上述，我们可以看到cookie又开始的作用并不是为了缓存而设计出来，只是借用了cookie的特性实现缓存

关于cookie的使用如下：

document.cookie = '名字=值';

关于cookie的修改，首先要确定domain和path属性都是相同的才可以，其中有一个不同得时候都会创建出一个新的cookie

Set-Cookie:name=aa; domain=aa.net; path=/ # 服务端设置  
document.cookie =name=bb; domain=aa.net; path=/ # 客户端设置

最后cookie的删除，最常用的方法就是给cookie设置一个过期的事件，这样cookie过期后会被浏览器删除

### localStorage

HTML5新方法，IE8及以上浏览器都兼容

### 特点

* 生命周期：持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的
* 存储的信息在同一域中是共享的
* 当本页操作（新增、修改、删除）了localStorage的时候，本页面不会触发storage事件,但是别的页面会触发storage事件。
* 大小：5M（跟浏览器厂商有关系）
* localStorage本质上是对字符串的读取，如果存储内容多的话会消耗内存空间，会导致页面变卡
* 受同源策略的限制

下面再看看关于localStorage的使用

设置

localStorage.setItem('username','cfangxu');

获取

localStorage.getItem('username')

获取键名

localStorage.key(0) //获取第一个键名

删除

localStorage.removeItem('username')

一次性清除所有存储

localStorage.clear()

localStorage 也不是完美的，它有两个缺点：

* 无法像Cookie一样设置过期时间
* 只能存入字符串，无法直接存对象

localStorage.setItem('key', {name: 'value'});  
console.log(localStorage.getItem('key')); // '[object, Object]'

### sessionStorage

sessionStorage和 localStorage使用方法基本一致，唯一不同的是生命周期，一旦页面（会话）关闭，sessionStorage 将会删除数据

### 扩展的前端存储方式

indexedDB是一种低级API，用于客户端存储大量结构化数据(包括, 文件/ blobs)。该API使用索引来实现对该数据的高性能搜索

虽然 Web Storage对于存储较少量的数据很有用，但对于存储更大量的结构化数据来说，这种方法不太有用。IndexedDB提供了一个解决方案

#### 优点：

* 储存量理论上没有上限
* 所有操作都是异步的，相比 LocalStorage 同步操作性能更高，尤其是数据量较大时
* 原生支持储存JS的对象
* 是个正经的数据库，意味着数据库能干的事它都能干

#### 缺点：

* 操作非常繁琐
* 本身有一定门槛

关于indexedDB的使用基本使用步骤如下：

* 打开数据库并且开始一个事务
* 创建一个 object store
* 构建一个请求来执行一些数据库操作，像增加或提取数据等。
* 通过监听正确类型的 DOM 事件以等待操作完成。
* 在操作结果上进行一些操作（可以在 request对象中找到）

关于使用indexdb的使用会比较繁琐，大家可以通过使用Godb.js库进行缓存，最大化的降低操作难度

## 二、区别

关于cookie、sessionStorage、localStorage三者的区别主要如下：

* 存储大小：cookie数据大小不能超过4k，sessionStorage和localStorage虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大
* 有效时间：localStorage存储持久数据，浏览器关闭后数据不丢失除非主动删除数据； sessionStorage数据在当前浏览器窗口关闭后自动删除；cookie设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭
* 数据与服务器之间的交互方式，cookie的数据会自动的传递到服务器，服务器端也可以写cookie到客户端； sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存

## 三、应用场景

在了解了上述的前端的缓存方式后，我们可以看看针对不对场景的使用选择：

* 标记用户与跟踪用户行为的情况，推荐使用cookie
* 适合长期保存在本地的数据（令牌），推荐使用localStorage
* 敏感账号一次性登录，推荐使用sessionStorage
* 存储大量数据的情况、在线文档（富文本编辑器）保存编辑历史的情况，推荐使用indexedDB

## 相关连接

* https://mp.weixin.qq.com/s/mROjtpoXarN--UDfEMqwhQ
* https://github.com/chenstarx/GoDB.js